

Research Report

KTC-15-18/SPR15-502-1F

DOI: <http://dx.doi.org/10.13023/KTC.RR.2015.18>

Identifying Significant Environmental Features Using Feature Recognition

Our Mission

We provide services to the transportation community through research, technology transfer and education. We create and participate in partnerships to promote safe and effective transportation systems.

© 2015 University of Kentucky, Kentucky Transportation Center
Information may not be used, reproduced, or republished without our written consent.

Kentucky Transportation Center

176 Oliver H. Raymond Building

Lexington, KY 40506-0281

(859) 257-4513

fax (859) 257-1815

www.ktc.uky.edu

**Research Report
KTC-15-18/SPR15-502-1F**

**IDENTIFYING SIGNIFICANT ENVIRONMENTAL FEATURES USING FEATURE
RECOGNITION**

By

Megan White, MA
Research Associate

Junfeng Zhu
Research Associate

Benjamin Blandford, PhD
Research Scientist

and

Ted Grossardt, PhD
Program Manager

Kentucky Transportation Center
College of Engineering
University of Kentucky
Lexington, Kentucky

In cooperation with
Kentucky Transportation Cabinet
Commonwealth of Kentucky

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the University of Kentucky, the Kentucky Transportation Center, the Kentucky Transportation Cabinet, the United States Department of Transportation, or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation. The inclusion of manufacturer names or trade names is for identification purposes and should not be considered an endorsement.

October 2015

| | | | |
|--|--|--|---|
| 1. Report No. KTC-15-18/SPR15-502-1F | 2. Government Accession No. | 3. Recipient's Catalog No | |
| 4. Title and Subtitle Identifying Significant Environmental Features Using Feature Recognition | | 5. Report Date October 2015 | |
| | | 6. Performing Organization Code | |
| 7. Author(s): Megan White, Junfeng Zhu, Benjamin Blandford, and Ted Grossardt | | 8. Performing Organization Report No. KTC-15-18/SPR15-502-1F | |
| 9. Performing Organization Name and Address Kentucky Transportation Center College of Engineering University of Kentucky Lexington, KY 40506-0281 | | 10. Work Unit No. (TRAIS) | |
| | | 11. Contract or Grant No. SPR15-502 | |
| 12. Sponsoring Agency Name and Address Kentucky Transportation Cabinet State Office Building Frankfort, KY 40622 | | 13. Type of Report and Period Covered | |
| | | 14. Sponsoring Agency Code | |
| 15. Supplementary Notes Prepared in cooperation with the Kentucky Transportation Cabinet | | | |
| 16. Abstract <p>The Department of Environmental Analysis at the Kentucky Transportation Cabinet has expressed an interest in feature-recognition capability because it may help analysts identify environmentally sensitive features in the landscape, including those relating to historic preservation, archaeology, endangered species habitat, and geology. LIDAR Analyst and Feature Analyst are a pair of geoprocessing software packages that have been developed by Textron Systems. Using this software, users can use LIDAR data to identify finely-scaled user-specified features. The software's automated feature extraction saves time that might otherwise be spent manually analyzing images and digitizing features.</p> <p>This report explores the capabilities and accuracy of this software by using LIDAR data to identify sinkholes throughout a small area in Kentucky. This report also discusses an alternative LIDAR-based geoprocessing methodology developed by the Kentucky Geological Society. The method relies on ArcGIS and Python scripting to identify sinkholes. The feasibility and applicability of these methodologies are compared, the workflow for each method is outlined, and the capabilities and limitations of each are noted. Sample results—the identification of sinkholes—from each methodology are presented. The research team found the batch processing capability built into LIDAR and Feature Analyst adequate and beneficial for smaller projects, such as projects that prioritize the extraction of buildings, trees, and forest regions.</p> | | | |
| 17. Key Words LIDAR, Feature Analyst, ArcGIS, Python, sinkholes, automated feature extraction | | 18. Distribution Statement | |
| 19. Security Classification (report) Unclassified | 20. Security Classification (this page) Unclassified | 21. No. of Pages 35 | 19. Security Classification (report) |

Table of Contents

| | | |
|-----|--|----|
| 1 | Executive Summary | 5 |
| 2 | Identifying Sinkholes Using Feature Analyst and LIDAR Analyst..... | 7 |
| 2.1 | The Software..... | 7 |
| 2.2 | Workflow and Capabilities | 8 |
| 2.3 | Sinkhole Models | 11 |
| 2.4 | Results..... | 16 |
| 3 | Identifying Sinkholes Using a Python-based ArcGIS Tool..... | 21 |
| 3.1 | Tool Installation..... | 21 |
| 3.2 | Tool Instruction..... | 23 |
| 3.3 | Tool Testing..... | 28 |
| 3.4 | Discussion..... | 29 |
| 4 | Discussion..... | 31 |
| 4.1 | Recommendations for Future Sinkhole Modeling..... | 31 |
| 4.2 | Other Modeling Features | 32 |
| 5 | Conclusions..... | 34 |
| 6 | Works Cited | 35 |

List of Figures

| | |
|---|----|
| Figure 1: Map showing features removed to create bare earth layer. Building footprints are outlined in black; tree crowns are outlined in blue. | 9 |
| Figure 2: Map demonstrating the clean-up process. | 10 |
| Figure 3: Study area within the Floyds Fork watershed. | 12 |
| Figure 4: Delineation of sinkhole training polygons begins with finding depressions in the topographic position index (TPI) raster. | 14 |
| Figure 5: Cleanup of initial results layer. | 18 |
| Figure 6: Verified sinkholes that were not returned by any of the models. | 20 |
| Figure 7: File tree for the ArcGIS tool. | 21 |
| Figure 8: Add Toolbox menu. | 22 |
| Figure 9: Add Toolbox dialog box. | 22 |
| Figure 10: The LIDAR sinkhole recognition tool shown in ArcGIS. | 22 |
| Figure 11: Script tool properties dialog box showing name of the script file. | 23 |
| Figure 12: Interface of script tool "Step 1a: Extract Depressions from DEM." | 24 |
| Figure 13: Example results of script tool "Step 1a: Extract Depressions from DEM." | 24 |
| Figure 14: Interface of script tool "Step 1b: Extract Depressions from LIDAR." | 25 |
| Figure 15: Interface of script tool "Step 2: Remove Non-Sinkholes Based on Surficial Features." | 26 |
| Figure 16: Example results of script tool "Step 2: Remove Non-Sinkholes Based on Surficial Features." | 27 |
| Figure 17: Interface of script tool "Step 3: Random Forests Classification." | 28 |
| Figure 18: Example results of script tool "Step 3: Random Forests Classification." | 28 |
| Figure 19: Test results showing: a) sinkholes previously mapped; and b) sinkholes identified from this tool. | 29 |

List of Tables

| | |
|--|----|
| Table 1: Setup details for five models tested. | 15 |
| Table 2: Results of the five tested models after the first learning pass and clean-up. | 17 |
| Table 3: Script tools and the corresponding python scripts. | 23 |

1 Executive Summary

Technological advances of the past decades have given rise to increasingly sophisticated photogrammetry and remote sensing techniques that have revolutionized terrain analysis. LIDAR has provided imagery of the earth's surface at an unprecedented level of detail. The Kentucky Transportation Cabinet (KYTC) has been acquiring larger amounts of high-quality LIDAR data for the state. Within this data there appear to be archaeological features and historical features that are better delineated than by current information sources. In Kentucky, archaeological features such as mounds and building foundations, natural features like sinkholes, and culturally or otherwise significant features like stone fences and cemeteries must be taken into account when road construction or maintenance activities are planned. When planning maintenance and when taking inventory of features, it is valuable to know the locations of engineered structures such as culverts.

Obtaining highly detailed imagery to conduct terrain analysis is easier than ever, but processing data can be difficult when dealing with large file sizes. Automated feature extraction programs offer one solution for analyzing large volumes of fine-grained data. These programs omit extraneous data components during processing and maximize resources for detecting features of interest. One of these applications, Feature Analyst, has the potential to assist users in efficiently identifying such features. Automated feature extraction saves time that might otherwise be spent manually analyzing images and digitizing features.

The purpose of this study was to summarize the functions and requirements of LIDAR Analyst and Feature Analyst. This report analyzes and discusses the process of extracting features of interest from LIDAR data. LIDAR Analyst transforms LIDAR data into products compatible with Feature Analyst and its supervised learning process. In Feature Analyst, customized models were created to extract sinkholes from the LIDAR Digital Elevation Models (DEMs). Model results and their relevance to KYTC are presented in this report, and included are speculations of Feature Analyst's performance in modeling other features of interest.

This report also details the development and application of a Python-based ArcGIS tool that identifies sinkholes across the landscape. While other types of geoprocessing innovations are able to perform landscape feature identification in a similar way as LIDAR Analyst and Feature Analyst, the python-based ArcGIS tool enables users to define the search criteria and to adjust the parameters to optimize feature recognition. ArcGIS is less user friendly than LIDAR Analyst, but it does have an extensive online support community and ample customization options for users familiar with Python. Both tools were applied to the identification of sinkholes within the Floyds Fork watershed, and the strengths and weaknesses were noted.

This research benefits the KYTC's Division of Environmental Analysis (DEA) by clarifying the positives and negatives of different methods of analyzing LIDAR data statewide. This information can assist the DEA in deciding how and where to use LIDAR data to aid its analysis of archaeological, environmental, and historic features. This research can be used to assess how extensively new GIS-based analyses of LIDAR data can be incorporated into DEA business processes over the long-term. The research establishes which landscape features can be identified more efficiently and accurately using LIDAR data, and notes improvement over current practices.

Performing more advanced extractions (i.e. buildings and trees) requires a much more complex work flow. Before attempting large-scale projects with LIDAR and Feature Analyst, Condor's distributed batch processing software would have to be successfully installed. While Textron tech support and programmers were willing to help, they were never able to provide a solution. However, the batch processing capability built in to LIDAR and Feature Analyst would be adequate for smaller projects. For projects that prioritize the extraction of buildings, trees, and forest regions, having LIDAR Analyst would be beneficial.

2 Identifying Sinkholes Using Feature Analyst and LIDAR Analyst

2.1 The Software

LIDAR Analyst offers automated extraction of 3D features, such as bare earth, buildings, and trees, from LIDAR (.las) data. There are user-friendly tools for cleaning up and otherwise refining extraction results. For further data manipulation capabilities, the program's 3D Viewer allows users to filter and edit point clouds directly. Feature Analyst is a feature extraction program that uses spectral information, spatial context, and physical characteristics of objects or areas to extract them from an image. User-generated training data, size and shape parameters, and templates that specify spatial context are the basis of the program's hierarchical learning process. After each feature is extracted, this can be increasingly modified until reasonable results are achieved.

Both LIDAR and Feature Analyst are produced by Textron Systems. They are third-party extensions of ArcGIS and can be used with ArcGIS 10 as well as with earlier versions. LIDAR Analyst can work directly with .las files, or with secondary LIDAR products like digital elevation models (DEMs). Feature Analyst learning models can use up to four bands of any raster image for each extraction. Both extensions come with built-in batch processing capabilities to manage extractions for large datasets. However, depending on the amount of data that requires processing, Textron recommends also obtaining Condor¹, a free, third-party distributed batch processing program that prioritizes and monitors processes to ensure optimal use of computing power. Textron Systems distributes a manual on acquiring and installing Condor, but the process is complicated and non-intuitive. Furthermore, tech support and engineers from Textron were never able to answer why Condor never worked properly on the KTC computers used to conduct the analysis described in this report. This problem was never resolved.

Because LIDAR and Feature Analyst are integrated into the ArcGIS interface, using them is very straightforward for people already familiar with Arc. Both extensions also come with a tutorial, an accompanying tutorial document outlining each step, and a reference manual. The tutorials present the basic work flow that most users would employ in almost every type of project. The tutorial document is exhaustive, describing software navigation, methodologies best suited for particular datasets, and settings and parameters. The reference manuals are lengthy and include descriptions of every tool the programs offer. Unlike with ArcGIS, online resources for LIDAR and Feature Analyst are extremely limited. There are no user forums, nor are there tutorials or other help documents online, other than those Textron provides with the software. The cost of a license includes tech support from Textron agents, who have proven to be knowledgeable,

¹ Available at <https://research.cs.wisc.edu/htcondor/>.

helpful, and in most cases, quick to respond to inquiries. The government pricing for Feature Analyst and LIDAR Analyst is identical. A node-locked license is \$3622.67, with an annual maintenance fee of \$1196.47. A floating (server-based) license is \$4529.47, with an annual maintenance fee of \$1435.77.

2.2 Workflow and Capabilities

LIDAR Analyst works directly with .las files. Typically, LIDAR analysis begins with users creating a bare earth image (i.e., DEM). This can be done by inputting a single or last return point cloud, a multiple-return point cloud, or first and last return rasters. As a part of the bare earth extraction process, LIDAR Analyst can also produce first and last return rasters and intensity images. The bare earth extraction dialog box offers pre- and post-processing options, including removing spikes and pits before processing, and smoothing the bare earth with a low pass filter to remove low height objects such as cars and shrubs. The user can adjust various parameters to control how the bare earth extraction treats tree and forest regions, ground slope thresholds, and interpolation. After the bare earth image has been created, LIDAR Analyst has a tool to manually edit results if there are irregularities.

After extracting bare earth, it is common to extract buildings, forests, and trees. There is a tool to extract buildings and a combined tool for extracting forest areas and tree crowns (Figure 1). Like the bare earth extraction tool, these have customizable parameters to let users specify extraction methods, characteristics of the features themselves, and post-processing options. For example, with the tree extraction tool dialog, the user can specify minimum tree height, typical tree height, and the minimum area that can be classified as a forest. In addition to these basic tools that most LIDAR Analyst projects require, the software has other tools for LIDAR data management, bare earth analysis, and feature editing and enhancement.



Figure 1: Map showing features removed to create bare earth layer. Building footprints are outlined in black; tree crowns are outlined in blue.

The first step in Feature Analyst’s workflow is creating a training dataset. Using an image of the study area, the user manually digitizes examples of the features to be extracted. These training polygons are input to the Learner, which houses the software’s adaptive learning component. The user inputs one or multiple bands of an image (often the bare earth raster). The size and shape of the search window and size thresholds for the features to be extracted is specified as well. The user can also opt to use a mask: either to exclude areas where it is known that features are irrelevant or non-existent, to indicate areas where knowledge of feature presence is needed, or to mark where the user knows features exist. Areas can be masked by vector layers, by pixel ranges, or by both. Masking reduces processing time and makes the clean-up process more efficient. With this information, the Learner returns features that meet the specifications. Once the first set of results are generated, a mark-up (clean-up) layer can be created to indicate examples of both correct and incorrect result features, thus refining or reinforcing the model’s framework (Figure 2). This step is repeated until results achieve the level of accuracy targeted by the user.

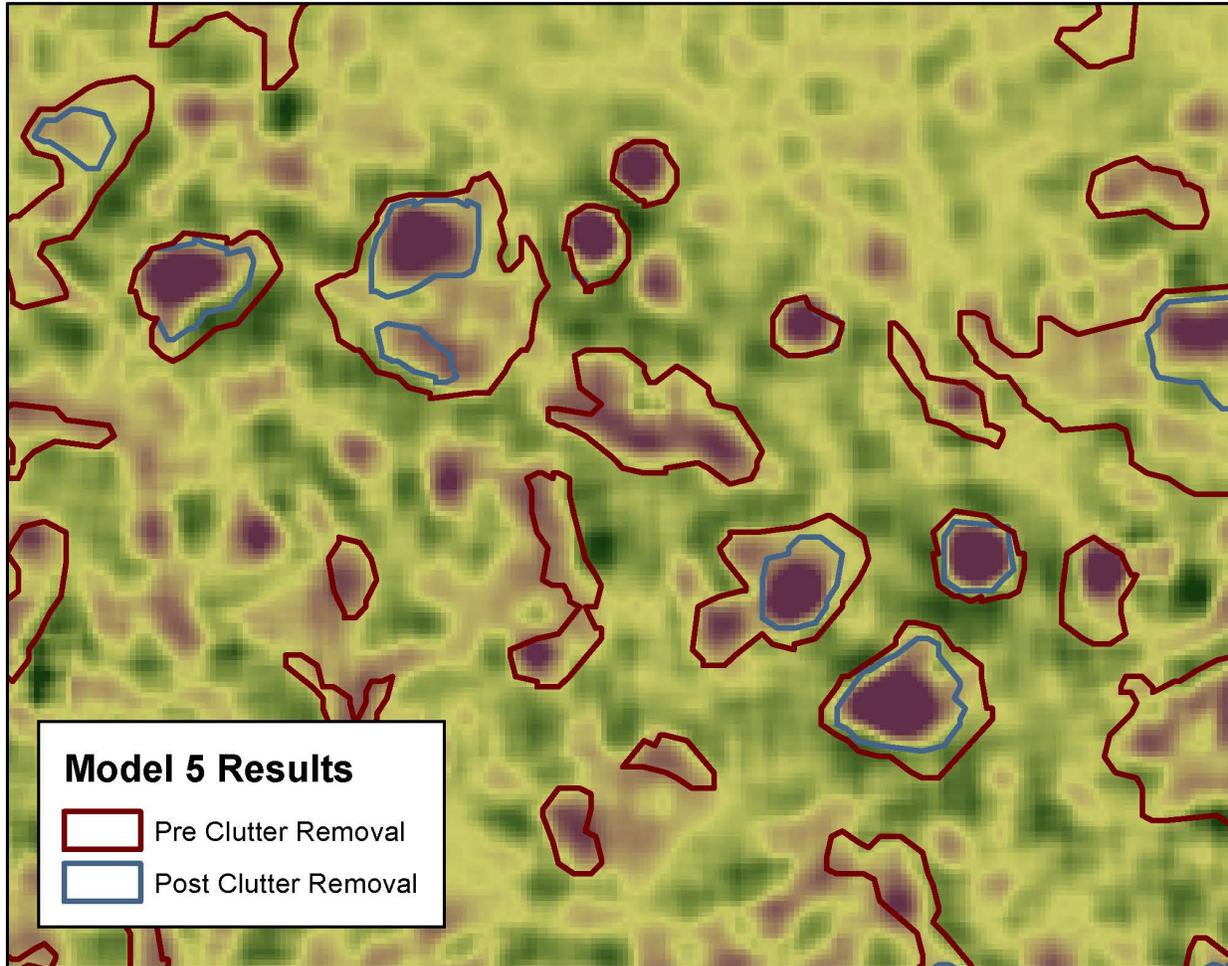


Figure 2: Map demonstrating the clean-up process.

To support the software’s feature extraction component, other tools are included to manipulate input images for easier extraction, and to edit and enhance the resulting features. A resampling tool enables the modification of raster resolution to better highlight features of interest. Image fusion lets users combine spectral information from multiple images into a single image. This is particularly useful when the number of image input bands exceeds the limit. For example, a Normalized Difference Vegetation Index could be fused to a bare earth image to highlight plant biomass alongside elevation. In terms of enhancing results, one tool allows result polygons to be rasterized and then eroded or dilated, to either sharpen polygon boundaries or connect them in a more network-like result feature. Another tool lets users further refine size parameters by specifying rules for aggregating features.

Both LIDAR and Feature Analyst rely on Automated Feature Extraction (AFE) files to track workflows and allow processes and models to be replicated on different datasets. AFE files are automatically saved in the same place as output files. Users can open these files and review the parameters set for specific processes or models, or they can import them into the batch

processing interface. If a project involves replicating a process on several files or running processes or models on very large areas, batch processing will be integral to LIDAR analysis and Feature Analyst workflows.

2.3 Sinkhole Models

A 16-tile, approximately 6.8 mi² area within the Floyds Fork drainage was selected to model sinkholes. This region is located on the Jefferson–Bullitt County border, approximately 5.5 miles east of Interstate 65, just northwest of Mount Washington (Figure 3). The 16-tile unit was chosen because it was sizeable enough to feature adequate heterogeneity, but also small enough to not require excessively long processing time. For the same model to be used across multiple areas, the files had to be merged into either a single .las file, or a single DEM, so the 16 tiles for each area were merged into single .las files using LAsTools. These .las files were used in LIDAR Analyst to create first and last return rasters. The first and last return rasters were used to generate bare earth DEMs. Generating the bare Earth DEM had the longest run-time. For this 16-tile, 6.8 mi² area, generating the DEM took just under two hours.

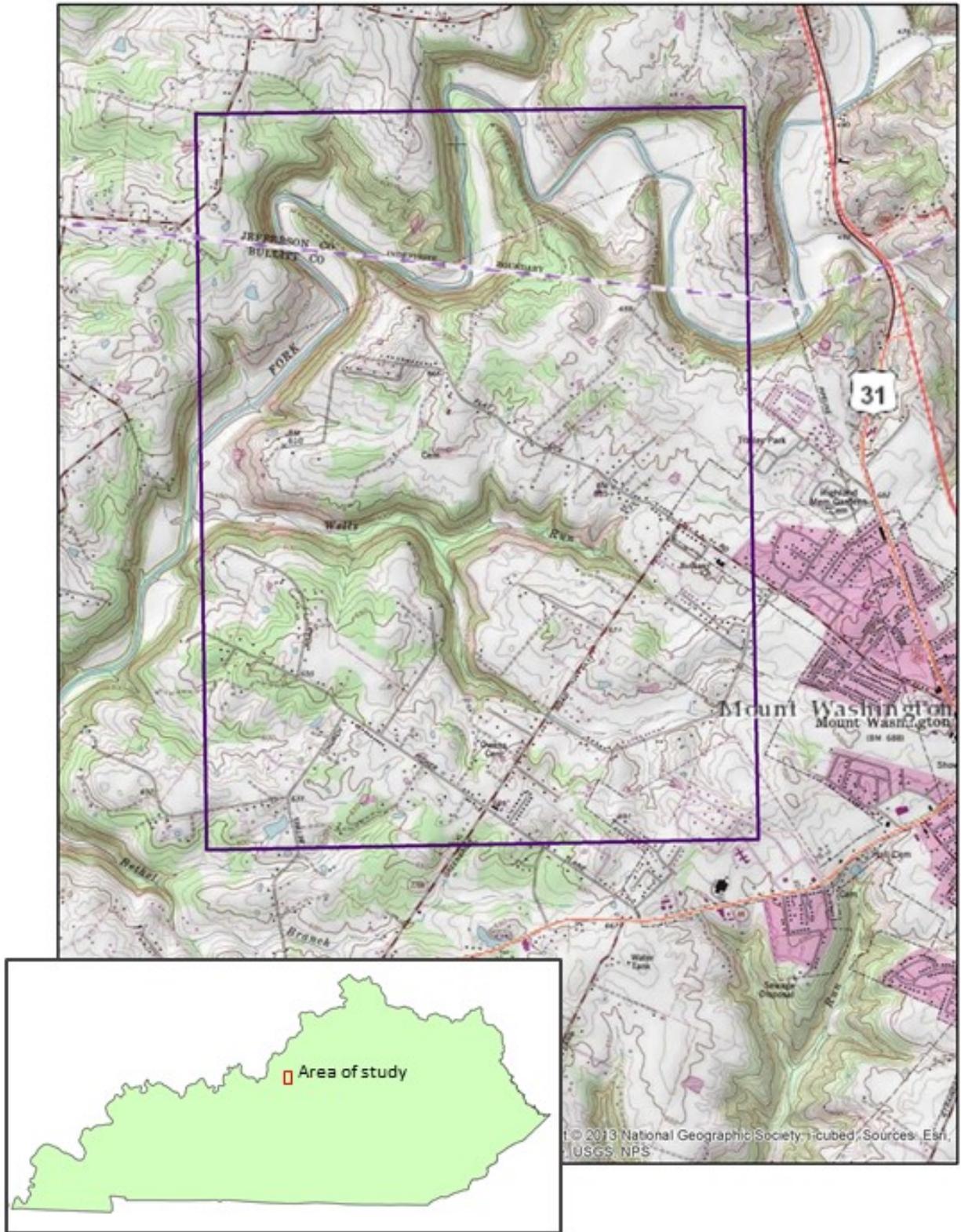


Figure 3: Study area within the Floyd's Fork watershed.

A high quality, representative training dataset is the basis of a good model in Feature Analyst. Because sinkholes were not readily apparent in DEMs, additional data were required to create training data. The bare earth DEM was used to create a topographic position index (TPI) raster and a slopeshade raster. The TPI was based on an algorithm that assigned a pixel value--calculated by finding the difference between its own elevation and the mean elevation of a user-defined neighborhood of surrounding pixels. A higher value meant a pixel was located at a higher elevation than nearby pixels; a lower value indicated it had a lower elevation than the surrounding pixels. A TPI raster can be created from a DEM using Land Facet Corridor Tools, a free ArcGIS extension available from Jenness Enterprises (Jenness et al. 2013). The slopeshade image was created with the Spatial Analyst slope tool.

Employing the methods of Doctor and Young (2013), the TPI was examined to identify circular or near-circular areas with significantly lower elevation values than surrounding areas. An area that met this requirement would then be investigated on the slopeshade raster to determine whether the feature appeared to be a closed or mostly closed depression. In non-forested and otherwise high visibility areas, ESRI basemap imagery was also referenced to verify the presence or absence of sinkholes, although in many cases this was not possible. Features meeting these criteria were delineated and used to populate the sinkhole training dataset. Figure 4 illustrates the process of creating training data.



Figure 4a

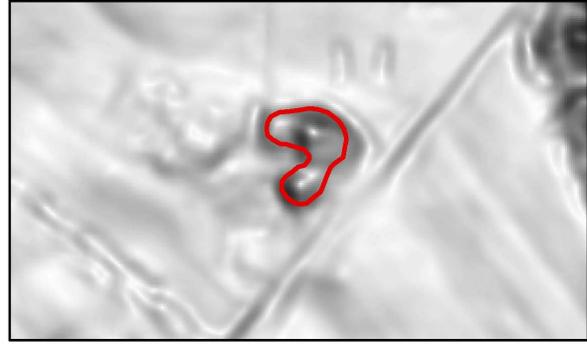


Figure 4b



Figure 4c

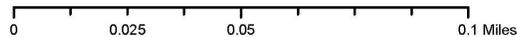


Figure 4: Delineating sinkhole training polygons begins with finding depressions in the topographic position index (TPI) raster (4a). Next, the slope raster is examined to verify an enclosed or semi-enclosed depression (4b). Finally, imagery is checked to validate previous findings. This will either rule the area out because of the presence of a feature that is not a sinkhole (sometimes the process picks up humanly constructed drainage features and small ponds), or confirm that the feature is, or likely is, a sinkhole. In the case of Figure 4c, a depression is not clearly visible from the imagery because of dense vegetation. However, this is an area of dense vegetation within a cleared field, so it is logical that a sinkhole could exist there.

Next, the training dataset was input into the Supervised Learning component of Feature Analyst. Aside from the training dataset itself, several other parameters were specified in the Supervised Learning dialog. The Feature Selector let users select from a list of basic feature types (Narrow Linear Feature, Wide Linear Feature, Small Manmade Feature, Natural Feature, etc.), each of which come with default model parameters. These defaults were most useful for initial learning passes and can be modified to refine the model. Input Bands provided the spectral information the model uses to locate features. Input Bands can be set to indicate reflectance, discrete pixel values, elevation, and texture. Because elevation is a key component to differentiate sinkholes

from the broader landscape, all the models included bands that indicate elevation. Some of the models also incorporated the slopeshade raster as a texture band.

Input Representation determined the shape and size of the search window that Supervised Learning used to determine if pixels were part of a feature of interest. Many of the Input Representation options were designed to extract linear features like sidewalks and streams and other built features such as buildings. The representation options geared toward finding natural features of varying sizes were Circle and Predefined Foveal. Trial and error proved the circle option to be the most reliable for identifying sinkholes. In all models, several areas were masked out of the learning process. Building footprints extracted in LIDAR Analyst were excluded from the models. 30 foot corridors on each side of rivers, streams, and roads were excluded as well to decrease false positives from ditches and other fluvial landforms. River and stream data were obtained from the National Hydrography Dataset (<http://nhd.usgs.gov/data.html>). The road centerline shapefile was obtained from the Kentucky Transportation Cabinet (<ftp://ftp.kymartian.ky.gov/trans/>). Water bodies, also obtained from the National Hydrography Dataset, were masked out as well.

In the Output Options tab of the Supervised Learning dialog, minimum and maximum feature area and smoothing tolerance can be specified. Size parameters for these models were derived from existing sinkhole shapefiles from the Kentucky Geological Survey (<http://www.uky.edu/KGS/gis/sinkpick.htm>) and from LIDAR Sinkhole Extraction tools (developed by Junfeng Zhu). Many models were tested for sinkhole extraction; Table 1 details the setup for five models, the results of which are discussed in this report.

Table 1: Setup details for five models tested.

| | Input Bands (Band Type) | Input Representation (Pattern Width in Pixels) | Minimum Area | Maximum Area |
|----------------|---|---|-------------------------|-------------------------|
| Model 1 | DEM, single band (elevation) | Circle (15) | 450 ft ² | 90,000 ft ² |
| Model 2 | Hillshade, band 1 (elevation) Hillshade, band 2 (elevation) Hillshade, band 3 (elevation) | Circle (15) | 450 ft ² | 90,000 ft ² |
| Model 3 | TPI (elevation) Slope (texture) | Circle (15) | 450 ft ² | 90,000 ft ² |
| Model 4 | DEM, single band (elevation) | Circle (15) | 450 ft ² | 30,000 ft ² |
| Model 5 | DEM, single band (elevation) | Pre-defined Foveal (9) | 450 ft ² | 30,000 ft ² |

Once the models were run, a mark-up (clean-up) layer was created to designate examples of correctly and incorrectly returned features. During this process, correct and incorrect features were selected and digitized, depending on the accuracy and precision of the model. Correct and incorrect features were identified by using the same methods employed to create training data. Samples of correct and incorrect features were selected in several areas of the image to ensure representative feedback for the model. Then, the same model was run on the edited clean-up layer to produce results defined by both the original model and by the clean-up specifications. When running this iteration of the model, an option was available to remove clutter. This option was based on shape attributes, and was always selected when running sinkhole models because it reduced the number of shapes that were long and linear, angular, and otherwise uncharacteristic of sinkholes.

On some occasions, running the model on the mark-up layer returned an empty feature class. In these instances, the mark-up layer was edited further to ensure that the model had an adequate number of correct features to learn from. Experimenting with this step showed that a ratio of no greater than 2:1 incorrect to correct features consistently returned results.

2.4 Results

Several factors affected the results, particularly the model accuracy. Given that there was no fieldwork conducted as part of this research, the research team was unable to validate results to evaluate accuracy. Second, because this was a relatively broad pilot study, certain criteria such as area of interest and sinkhole characteristics (e.g., size, or sinkhole density) in a given area were left unspecified. Because the Learner in Feature Analyst operated on a holistic set of detailed parameters, its accuracy may have declined when the features to be extracted had more variable characteristics and occurred across a broader variety of landscapes. Likewise, if the features to be extracted met certain specifications (i.e., occur within 100 feet of a road or had areas of greater than 1000 feet²), the Learner's accuracy and precision may have increased. That said, there are factors in this study that helped verify the accuracy, and at least the logic, of the models' results. 12 verified sinkholes in the area of study were used as benchmarks for the extracted features. Given the relatively small area of study, researchers visually examined the results alongside detailed elevation layers, slope layers, and imagery. This was done to qualitatively assess accuracy.

Table 2 summarizes the results of the five models after the first learning pass as well as after clean-up. The table lists the total number of features returned by the models after each learning pass, the number of features that intersect verified sinkhole polygons, and training data polygons. The result table indicates only that result polygons intersect verified sinkhole polygons or training polygons. In the majority of cases, this meant the result polygons loosely or closely matched the outlines of the verified or training polygons. In a few cases, however, result polygons intersected verified or training polygons without capturing the depression feature.

While training data polygons were carefully selected and digitized and were certainly depression features in the landscape, there cannot be positive identification that they are sinkholes without field verification.

Table 2: Results of the five tested models after the first learning pass and clean-up.

| | First Learning Pass | | | Second Learning Pass | | |
|---|---------------------|--|---|----------------------|--|---|
| | # Returned Features | # Features Intersecting Verified Sinks (/12 total) | # Features Intersecting Training Data (/29 total) | # Returned Features | # Features Intersecting Verified Sinks (/12 total) | # Features Intersecting Training Data (/29 total) |
| Model 1 (Circle 15, DEM) | 2847 | 10 | 20 | 898 | 9 | 20 |
| Model 2 (Circle 15, Hillshade) | 5090 | 9 | 26 | 632 | 4 | 8 |
| Model 3 (Circle 15, TPI/slope) | 5977 | 10 | 29 | 33 | 2 | 4 |
| Model 4 (Circle 15, DEM, 30,000 max area) | 2837 | 10 | 20 | 481 | 9 | 18 |
| Model 5 (Pre-def Foveal 9, DEM) | 2748 | 6 | 22 | 492 | 5 | 20 |

Initial learning pass results of Model 1 were surpassed by Model 3, but Model 1 had the greatest number of features intersecting verified sinkholes and training polygons after the second learning pass. Model 2 had comparable results to Models 1 and 3 after the first learning pass, but its accuracy deteriorated following clean-up. Model 3, which used the TPI and slope rasters as input returned the highest number of features intersecting both verified and training polygons after the first learning pass, however, its accuracy also fell dramatically after clean-up. This was unexpected, as the TPI and slope inputs have generally performed better than all the others. Even with various versions of clean-up parameters, Model 3 had very poor post-clean-up results. Model 4 had the fewest returned features after the initial learning pass. It had 10 fewer returned features than Model 1, and all its other features were identical to those in Model 1. Changing the

maximum allowable area had minimal impact because very few features above 30,000 ft² were extracted to begin with. Model 5 performed relatively poorly by these metrics in the first learning pass. After clean-up, the number of features in Model 5 that intersected verified sinks fell in the middle of the other model results, but features that intersected training polygons tied Model 1 for the highest.

Shape was also an important criterion for measuring model performance. After the first learning pass, all the models performed relatively well at capturing verified polygons and training polygons, but some were more precise than others. Model 2 required more digitizing because, even in cases where it captured sinks, they often had erratic boundaries. Models 1 and 3 had the most accurate shapes in the first learning pass. With the option of removing clutter based on shape attributes, this precision will carry over into the second learning pass even if overall accuracy declines. This is exemplified by Model 5 in Figure 5.

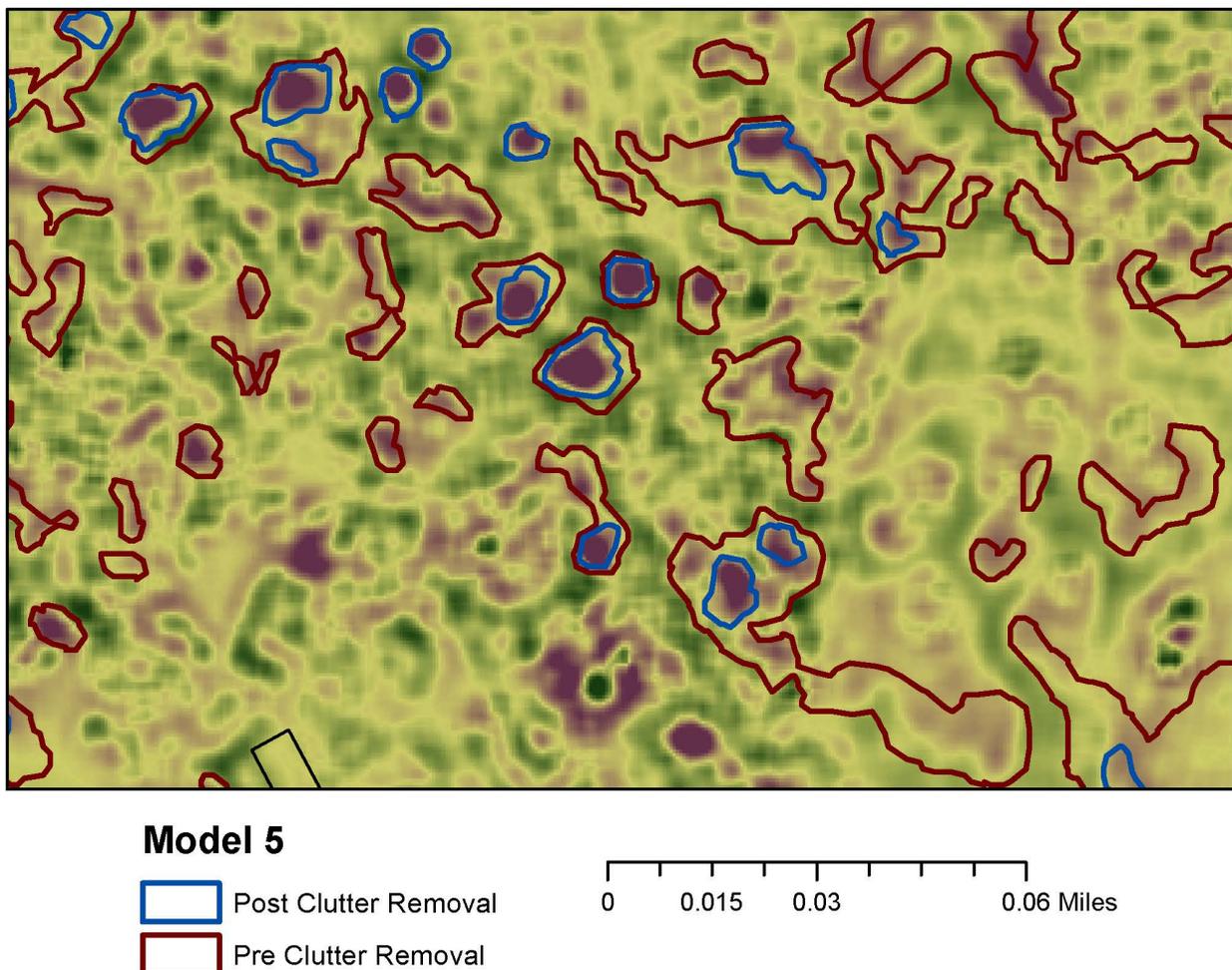


Figure 5: Cleanup of initial results layer. This refines model performance.

Aside from coincidence with definite or probable sinkholes, another important aspect of model performance was the number of false positives it returned. Even after clean-up, most of the

models still produced an extremely high number of returned features for a 6.8 mi² area. It was obvious that, while many were depressions, most of them were not sinkholes. Further clean-up and learning passes could cut down on the number of false positives returned by a model, but because of the large size range and variety of shape possibilities sinkholes have, false positives will inevitably occur. Many of the false positives were easy to rule out by visually scanning the imagery. Culverts were often extracted, as were other drainage features near houses. In some instances, the building polygons extracted by LIDAR Analyst were not exact, so models sometimes picked up corners or edges of buildings as sinkholes. Adding a buffer to the building polygons for masking could alleviate this problem. In cases where fieldwork will be performed irrespective of model results, false positives may not be problematic. For example, ground truthing 15 sites would not take much longer than three sites as long as the study area had a limited extent.

The problem of false negatives was more difficult to correct. This issue is best illustrated by looking at the verified sinkholes the models consistently failed to detect. For example, after the second learning pass, none of the models picked up the verified sinkholes seen in Figure 6a or 6b. However, all the models spotted the verified sinkholes in Figures 6c and 6d. (In figure 6c, Model 3 did not align accurately with the verified sink, but it did pick up a small section.) It is clear that the verified sinkhole polygons in Figures 6a and 6b do not have the same signature in the TPI (or in the slope or hillshade rasters) as the verified sinkholes in 6c and 6d, or as the other possible sinkholes viewable in the screenshots. The sinkholes appear lumpier in the TPI, slope, and hillshade images, with more subtlety in their elevation variation compared to the dark, closed, circular looking depressions that the models detected. The irregularity of sinkholes can make them difficult to model, especially without narrowing the search to a more specific type of sink. It is possible that particular kinds of sinkholes, like cover collapse, might be easier to extract than a broad classification of sinkhole. It is also possible that the likelihood of the models picking out anomalous features would remain high.

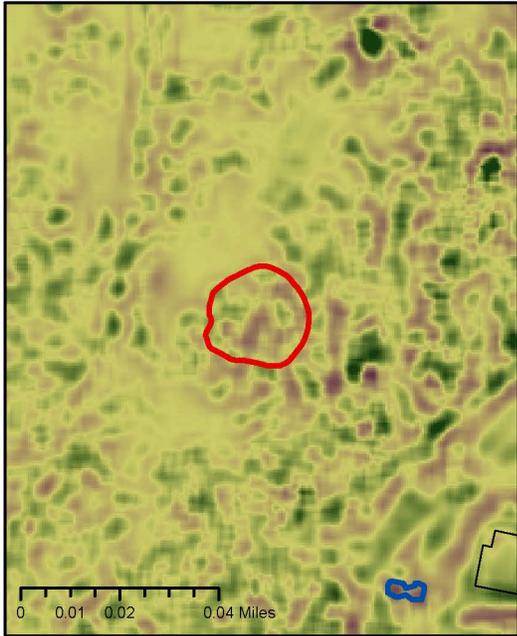


Figure 6a



Figure 6b

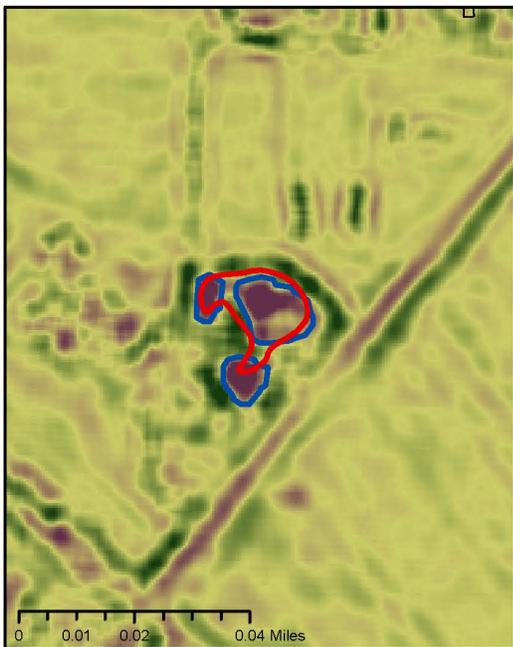


Figure 6c

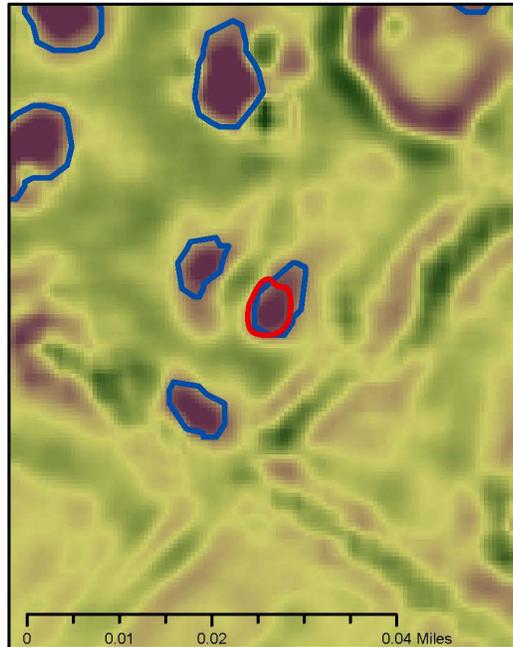


Figure 6d

- Verified Sinkholes
- Model 1, Post Clean-up

Figure 6: The polygons in Figures 6a and 6b represent the only verified sinkholes that were not returned by any of the models. Compared to other examples of sinkholes, their elevation change is much less dramatic. Figures 6c and 6d show the only verified sinkholes returned by all models after the second learning pass. Both exhibit concentrated areas of lower elevation than the surrounding area.

3 Identifying Sinkholes Using a Python-based ArcGIS Tool

Sinkholes are a common geologic hazard that impact road construction in Kentucky. The purpose of this ArcGIS tool is to automatically recognize possible sinkholes from LIDAR data, thus saving time and effort in the field. This tool was developed using python scripts and works with ArcGIS 10.1 and above. The tool is provided as a compressed file, LIDARSinkholeRecognitionTool.zip, which contains an ArcToolBox file (.tbx), four python script files (.py), and four files needed for an automatic classification method. The files are organized as follows (Figure 7):

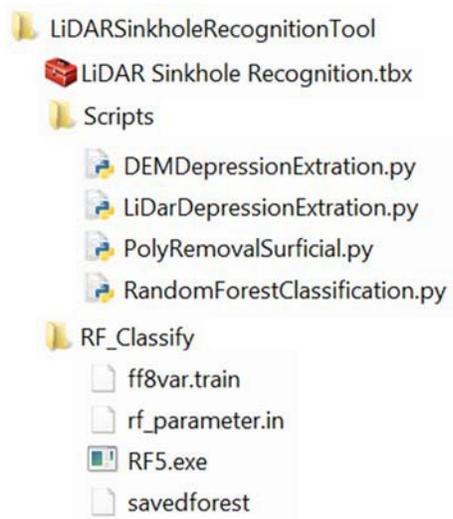


Figure 7: File tree for the ArcGIS tool.

This tool partitions the sinkhole extraction process into three steps. The first operation extracts surface depressions as polygons from either a DEM or LIDAR point clouds. The second step removes non-sinkhole polygons using information on other surface features, such as hydrology and transportation. The third operation applies a machine learning algorithm, called Random Forests, to identify possible sinkholes from the remaining polygons.

3.1 Tool Installation

To install the tool, extract LIDARSinkholeRecognitionTool.zip to a specified location. After the extraction is completed, compare the file tree with Figure 1 to ensure that all files are present.

The next step of installation is importing the tool into ArcMap. In ArcMap, open ArcToolbox and right-click the topmost item, ArcToolbox, to open a menu (Figure 8).

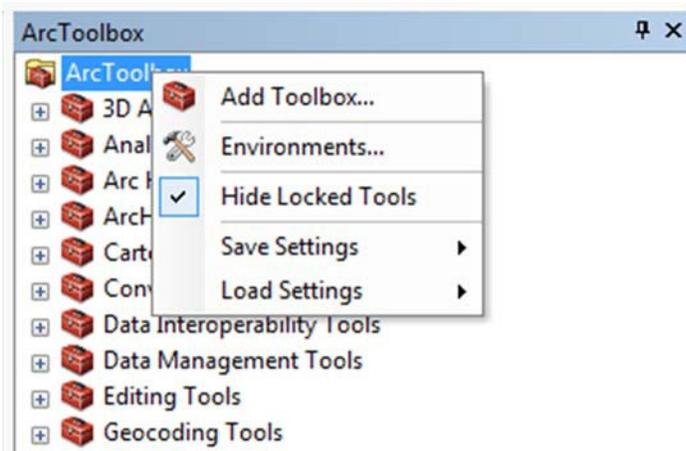


Figure 8: Add Toolbox menu.

Select “Add Toolbox” to open “Add Toolbox” dialog box (Figure 9).

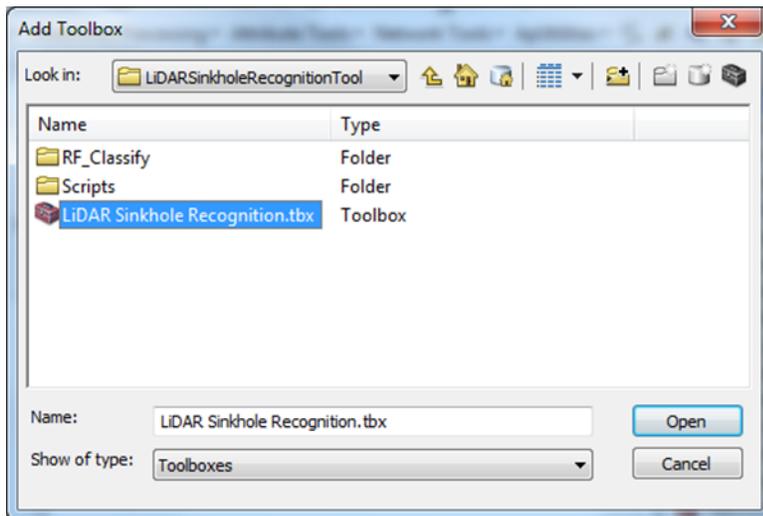


Figure 9: Add Toolbox dialog box.

Navigate to the location where the tool is located, select “LiDAR Sinkhole Recognition.tbx” and click “Open”. In a few seconds, a new toolbox, “LiDAR Sinkhole Recognition,” appears in ArcToolbox. Click the “+” next to the new toolbox. This reveals four script tools under the “LiDAR Sinkhole Recognition” (Figure 10).

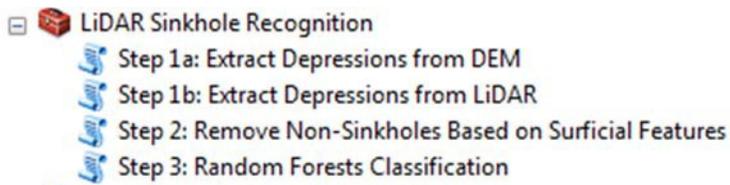


Figure 10: The LiDAR sinkhole recognition tool shown in ArcGIS.

The toolbox is now ready to go. Because each script tool relies on a python script file that is not embedded in the tool, make sure that each one is linked to its corresponding script file (Table 3). To check, right-click any of the script tools—choose “Properties...”, and click “Source” tab. The script file will appear with its full path under “Script File” (Figure 11). If there are incorrect links fix them using the button to the right.

Table 3: Script tools and the corresponding python scripts.

| Script tool name | Python script name |
|--|-------------------------------|
| Step 1a: Extract Depressions from DEM | DEMDepressionExtraction.py |
| Step 1b: Extract Depressions from LIDAR | LIDARDepressionExtraction.py |
| Step 2: Remove Non-Sinkholes Based on Surficial Features | PolyRemovalSurficial.py |
| Step 3: Random Forests Classification | RandomForestClassification.py |

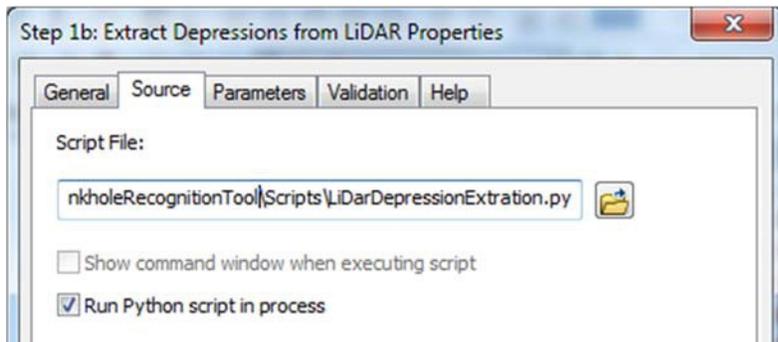


Figure 11: Script tool properties dialog box showing name of the script file.

3.2 Tool Instruction

The tool partitions sinkhole recognition into three steps. Each one is described sequentially below.

Step 1.

Begin by extracting surface depressions as polygons. Either a DEM or LIDAR point cloud dataset is needed. Use “Step 1a: Extract Depressions from DEM” if a DEM is provided. This is common given that LIDAR-derived DEMs are much easier to acquire than LIDAR point clouds for the state of Kentucky. To extract surface depressions from a DEM, prepare a DEM and have a File Geodatabase available to store outputs. Double-click the “Step 1a: Extract Depressions from DEM” under the “LIDAR Sinkhole Recognition” to bring up the dialog box (Figure 12).

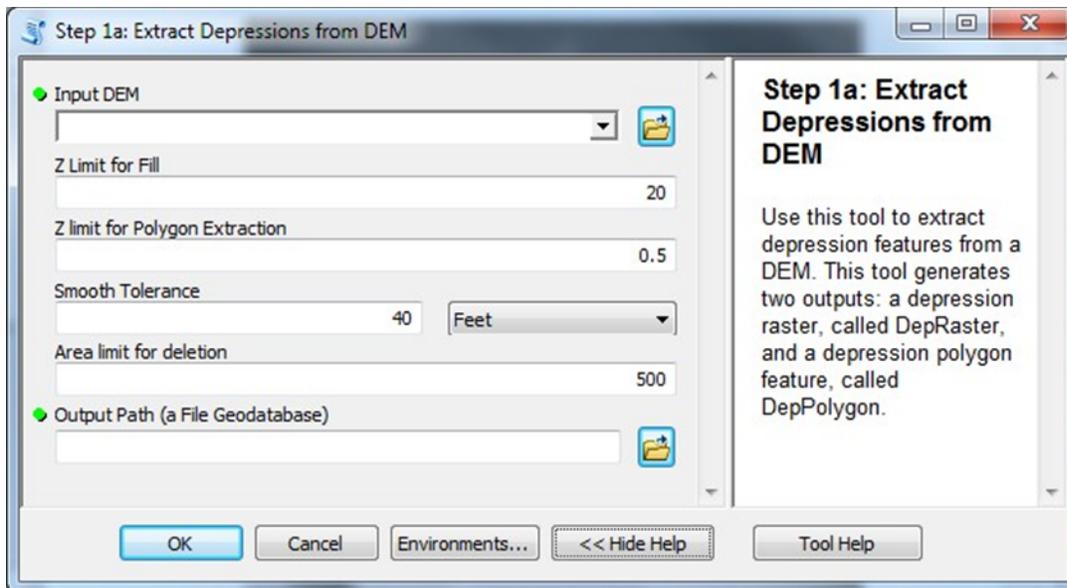


Figure 12: Interface of script tool "Step 1a: Extract Depressions from DEM."

Detailed help is available for all script tools. To obtain help for an input item in a tool dialog box, click the dialog area next to the item and the help will appear to the right. Input items with a green mark next to them are required, and others are optional with default values provided. These values can be adjusted from defaults to fit the needs of a particular analysis. Once all inputs are entered, click "OK" and the tool will run. After the tool has executed, it will generate two output items — a depression raster, called DepRaster, and a depression polygon feature, named DepPolygon. Both are stored in the user-specified File Geodatabase. Figure 13 shows an example of the results with DepRaster and DepPolygon plotted atop a DEM's hillshading.

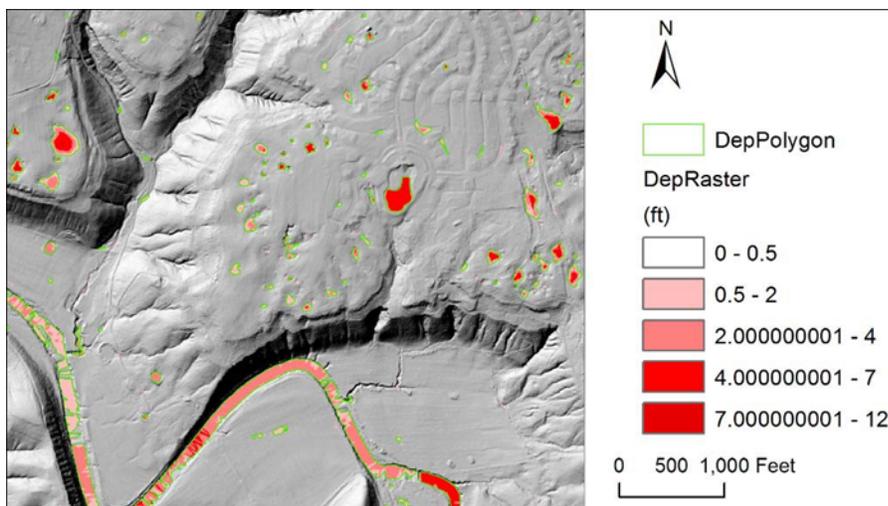


Figure 13: Example results of script tool "Step 1a: Extract Depressions from DEM."

The script tool, “Step 1b: Extract Depressions from LIDAR,” can be used to extract depression polygons if LIDAR point clouds are available. To use this tool, prepare a LAS Dataset (.lasd) and have a File Geodatabase available to store outputs. A LAS Dataset is used in ArcGIS to process LIDAR data. Refer to ArcGIS Help for detailed instruction on creating a LAS dataset from LIDAR data. Double click the “Step 1b: Extract Depressions from LIDAR” under the “LIDAR Sinkhole Recognition” to bring up the dialog box (Figure 14).

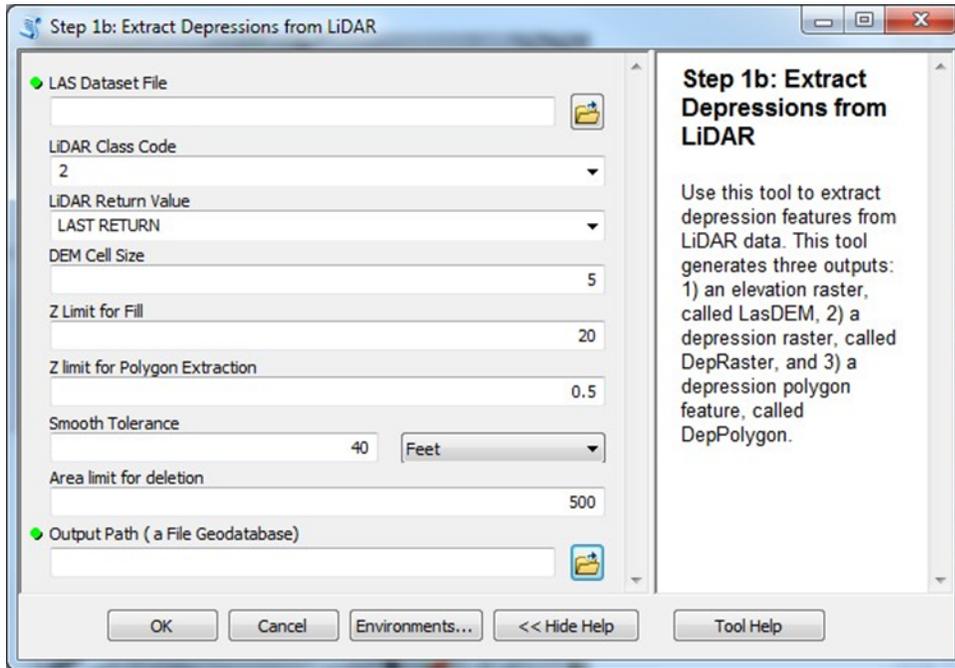


Figure 14: Interface of script tool "Step 1b: Extract Depressions from LIDAR."

Compared to the script tool, “Step 1a”, this tool adds an additional function that creates a DEM from LIDAR data. Once the DEM is created, users follow the same steps. Enter all the required inputs (an LAS dataset and a File Geodatabase in this case) and modify optional inputs as needed. Then, click “OK” to run the tool. After executing, the tool will generate three outputs: 1) an elevation raster, called LasDEM; 2) a depression raster, called DepRaster; and 3) a depression polygon feature, called DepPolygon. All outputs are stored under the File Geodatabase the user specifies.

Step 2.

Many of the depression Polygons (DepPolygon) extracted during Step 1 are not sinkholes. Non-sinkhole polygons may be ponds, stream channels, or man-made structures. The script tool, “Step 2: Remove Non - Sinkholes Based on Surficial Features,” uses existing coverages on surface features to remove non-sinkhole polygons. To use this tool, prepare coverages for rivers/streams, lakes/ponds, roads, and land uses. Figure 15 represents the script tools interface. Refer to the help for the script tool for instruction and possible sources for surficial coverages. If

the user has information about other surficial features, use the “Input Optional Filter Features” box to add an additional feature into the process. Notice that the land uses must be in raster format, while all other coverages should be polygons or polylines. The land use raster should be cropped to the area of interest to avoid unnecessary computing time.

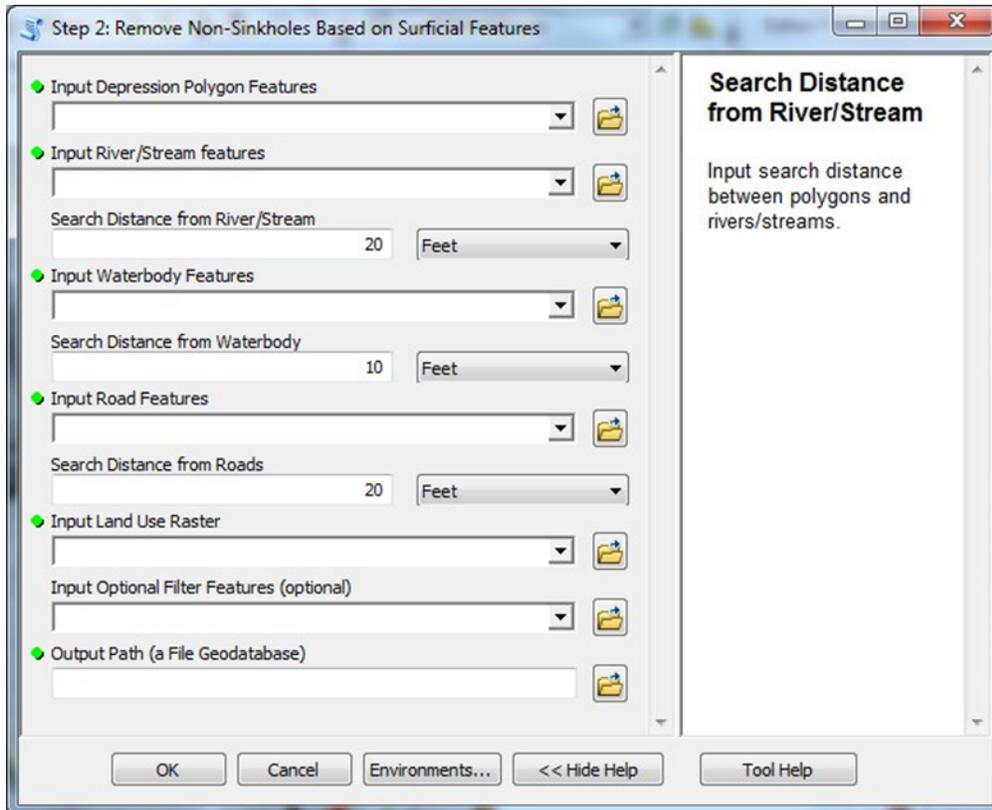


Figure 15: Interface of script tool "Step 2: Remove Non-Sinkholes Based on Surficial Features."

This tool will generate a new feature, “DepPolygonFiltered”, which contains the polygons that are not filtered out by the surficial features. Figure 16 illustrates example results from this script, showing some of the depression polygons have been filtered out (those shaded green).

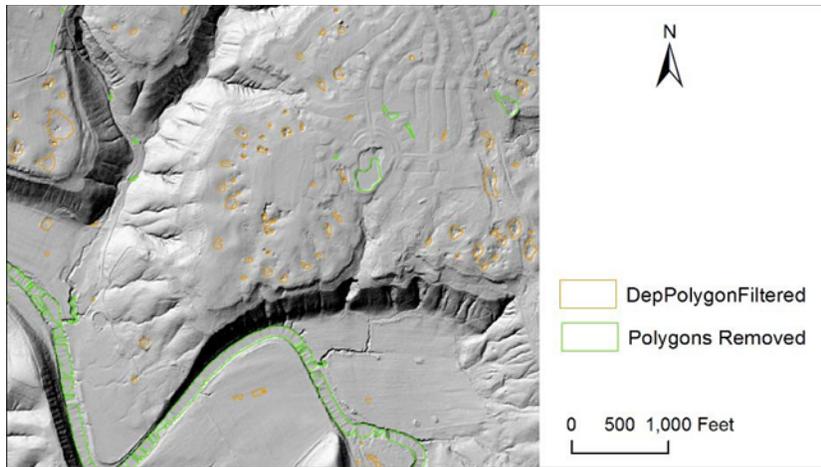


Figure 16: Example results of script tool "Step 2: Remove Non-Sinkholes Based on Surficial Features."

Step 3.

In this step, a machine learning algorithm called “Random Forests” is applied to classify the remaining polygons into sinkholes and non - sinkholes. The Random Forests (RF) algorithm was developed by Leo Breiman (2001) and Adele Cutler. The RF algorithm constructs multiple classification trees (i.e., forming forests) using a training dataset. New data are then classified by combining results from all classification trees through a voting process. A robust training dataset is essential for accurate classification. This tool adopts a LIDAR sinkhole dataset for Floyds Fork watershed (Zhu et al., 2014) as its training dataset. The Floyds Fork sinkhole dataset was created using a depression extraction process identical to the Script Tool 1b, but the polygons were classified manually through visual inspections. Field validation indicated the dataset’s accuracy was between 80% and 93%. To run the RF classification, double click the script tool “Step 3: Random Forests Classification.” The interface is shown in Figure 17. This script tool requires the depression polygons (i.e., the depression raster generated in step 1a or step 1b) and the location of the RF program. The folder should be called “RF_Classify” under folder “LIDARSinkholeRecognitionTool.” (Refer to Figure 1 for more details). This tool will generate two polygons features, "DepPolyFiltered_RF_Sink" for possible sinkholes and "DepPolyFiltered_RF_NonSink" for likely non-sinkholes. Sample results from this script tool are shown in Figure 18.

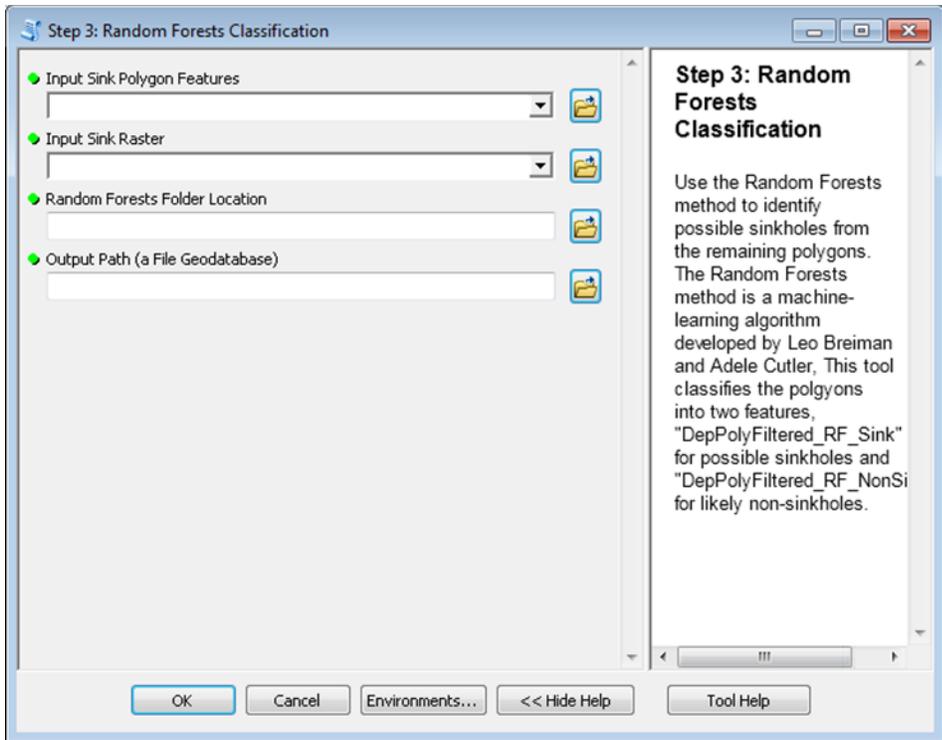


Figure 17: Interface of script tool "Step 3: Random Forests Classification."

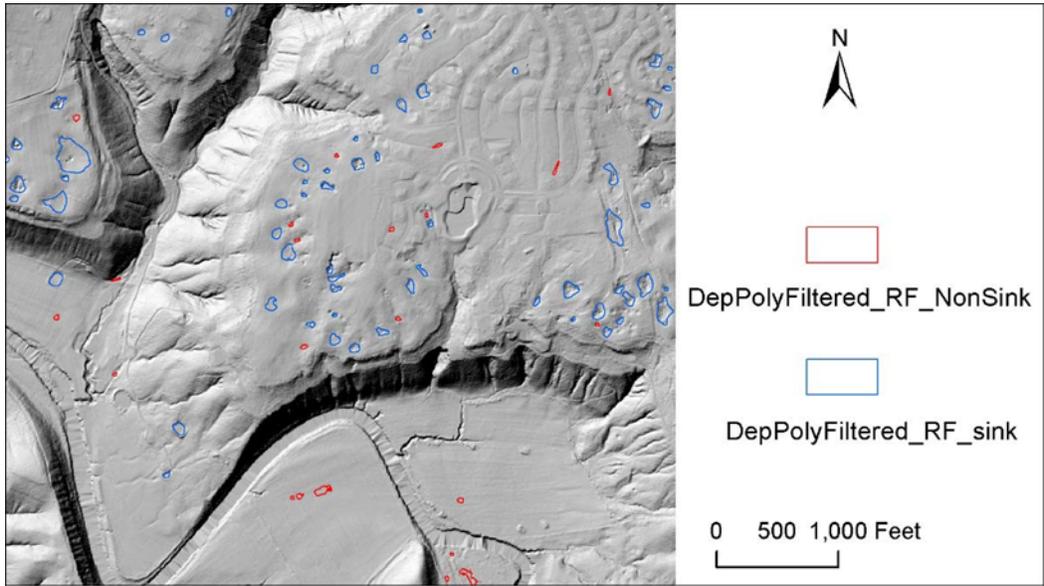


Figure 18: Example results of script tool "Step 3: Random Forests Classification."

3.3 Tool Testing

The LIDAR sinkhole-recognition tool was tested on a 1.72 square mile area inside Floyds Fork watershed, where 66 sinkholes (Figure 19a) have been previously mapped using LIDAR (Zhu et

al., 2014). The tool identified 83 possible sinkholes in this setting (Figure 19b). All 66 of the previously mapped sinkholes were identified by this tool as sinkholes. Inspecting the 2012 6-inch aerial images revealed the 17 false positives encompassed a variety of features.

Some of the false positives appeared on top of residential building roofs, which were likely artifacts within the LIDAR data. Some were seen as constructed drains. One appeared above a water-filled pond. This pond was apparently excluded from the NHD dataset. A few others appeared in forested area and appeared to be possible sinkholes that were missed during earlier mapping efforts.

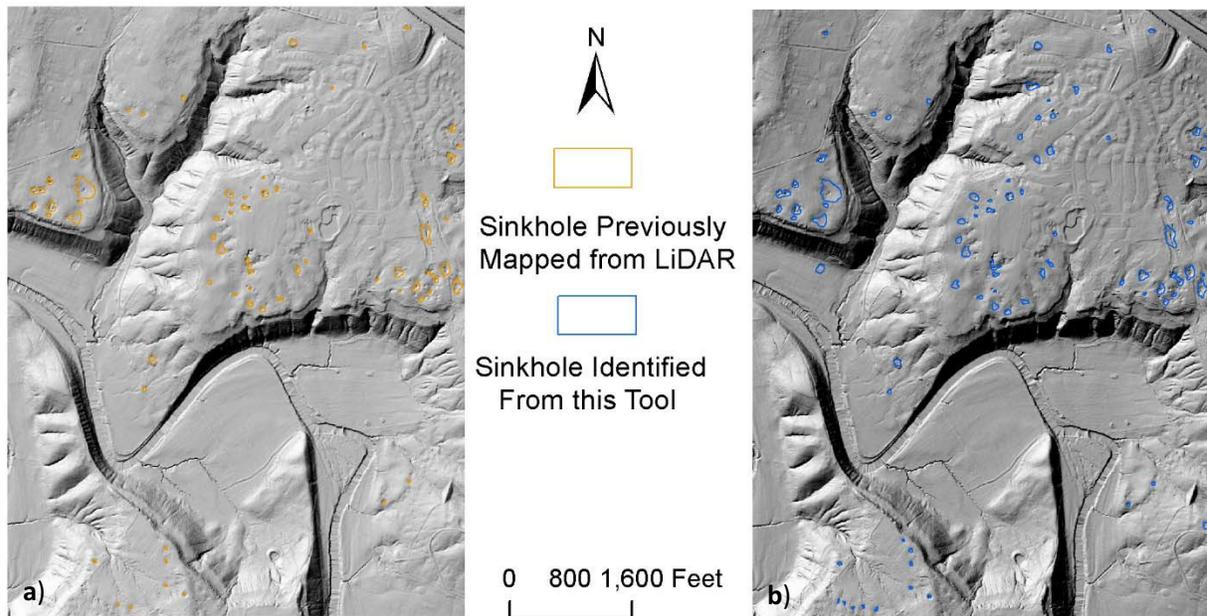


Figure 19: Test results showing: a) sinkholes previously mapped; and b) sinkholes identified from this tool.

3.4 Discussion

A Python-based ArcGIS tool was developed to recognize karst sinkholes using LIDAR data. This tool was compatible with ArcGIS 10.1 and above as an ArcToolbox. A test case demonstrated its promise at recognizing sinkholes from LIDAR data. When using this tool, users should also pay attention to the following:

1. Although the test case demonstrated that this tool efficiently detects sinkholes, it also misidentified other features as sinkholes (i.e., false positives). In all likelihood, the tool will miss some sinkholes when applied to other areas. Yet, this tool can potentially produce significant time and labor savings, reducing the amount of fieldwork required to pinpoint sinkholes. But it is critical to remember that this tool cannot serve as a substitute

for the human eye. Results generated from this tool should be manually checked and verified in the field to ensure accuracy.

2. This tool will function most effectively when applied to limited areas. We recommend that DEM file size not exceed 100 Mb.
3. The team derived the training dataset from polygons of 500 square feet or larger. Applying this training dataset to find smaller sinkholes may produce less accurate results.

4 Discussion

4.1 Recommendations for Future Sinkhole Modeling

There are several options for improving the performance of sinkhole models. The first option is to continue tweaking model parameters in Feature Analyst until desirable results are achieved, ideally with the same model for different areas, to ensure some degree of transferability for the model. Experimenting with different training datasets, using different clean-up methods (i.e., doing more or less thorough clean-up of results), and selecting different ratios of correct to incorrect features would be a great place to start. This would grant the user a clearer picture of how modifying settings affects results, and subsequently would allow for more control.

When researchers closely examined the imagery during the clean-up process, it became apparent that some of the vector layers used for masking did not line up precisely with the actual features on the ground. This was particularly true for flowlines and bodies of water. Several of these polygons were slightly offset from the actual features. Also, using a universal 30-foot buffer on the roads shapefile produced different effects on small two-lane roads, interstates, and larger highways. This is because the buffer extended out on either side of the road centerline, which was of negligible width. Using the highest resolution data available for masking likely would not make a significant difference in the results of this study, but would slightly reduce the amount of clean-up required. In other cases, depending on the features being modeled and their proximity to masked layers, using high resolution data for masking could be critical.

It might be worthwhile to consider the size and shape attributes of different types of sinkholes (i.e., solution, subsidence, cover-collapse) and what distinguishes them from one another in DEMs. If significant differences could be discerned between these broad classes of sinkholes in the DEMs, creating different models that reflect the classes' individual characteristics would eliminate many false positives and reduce the excessively high number of returned features seen in the models.

Conducting fieldwork to verify training data and results would prove invaluable for refining the models. First, it would provide the essential function of confirming or disconfirming that training polygons are verified sinkholes. Second, it would be extremely beneficial to creator of the models to see what false positive results look like on the ground; then, better steps to eliminate them could be taken. It is possible that a mask (based on slope, pixel values, and other attributes) could be created to exclude the majority of landform features that are false positives. However, discerning the characteristics this mask might have would likely require spending considerable time in the field checking the several false positive features returned by the models.

For any large-scale sinkhole modeling projects (involving around fifty or more tiles), careful thought should be put into how processing will be set up. Experience from this project suggested that processing times and capacity to model sinkholes were more feasible on an area- by-area, or project-by-project basis. It would be possible to model sinkholes for a region of Kentucky, or for the entire state; however, it would be a massive undertaking. Even processing an area of 100 tiles (approximately 40 mi²) would require a plan for subdividing the data, either to manually create multiple DEMs to encompass the entire area, or for specifying to the batch processing tool how to handle the dataset. Using the batch processing tool can be convenient, but it will also require greater organization and forethought than running one process on one dataset at a time.

4.2 Other Modeling Features

With an increasing amount of LIDAR data available for much of the state of Kentucky, further exploring Feature Analyst's modeling capabilities for landscape features other than sinkholes would be worthwhile. The software has components — unexplored in this study — geared toward extracting larger land cover features, such as wetlands. The Feature Selector in the Supervised Learning dialog has options for Land Cover Feature and Water Mass Feature. Either of these Feature Selector options could provide a good starting point to extract wetland areas, depending on the time of year and degree of saturation of the soil. Choosing one of these (from the list of six other options) automatically sets the other Supervised Learning parameters to optimize potential for extracting these types of features.

LIDAR and Feature Analyst offer several possibilities for extracting trees and forest areas from LIDAR-derived DEMs. As described previously (and illustrated in Figure 1), LIDAR Analyst can automatically extract individual trees and forest areas and symbolize them with points and polygons, respectively. This tool lets users specify settings for minimum tree height, typical tree height, and the size at which to aggregate a forest area. The tool also provides two different extraction methods, one with a fixed window search that is better for more densely forested regions, and another with a variable search window, which can be used to obtain more accurate crown widths. For further control over tree and forest parameters, there are settings that can be adjusted during the bare earth extraction, before using the tree and forest extraction tool. Bare earth extraction lets the user specify the minimum ground slope for tree regions, the minimum texture, minimum difference between point returns, sampling frequency, and minimum forest size. These options for bare earth and tree extraction are valuable because they require minimal user input, unlike the training and clean-up processes in Feature Analyst.

Culverts — specifically stone culverts — are another feature of interest on transportation projects that Feature Analyst could be used to model. Some work with culvert modeling was done in this project, but the work was not sufficient to determine whether the software can reliably extract these features. However, researchers gained insights into the initial steps of modeling. Creating training data for the culvert models was more straightforward than for the

sinkhole models because many of the culverts were clearly visible in the ArcGIS basemap imagery. Training polygons can be digitized by identifying intersections between stream and road vector data, and then imagery can be examined to check for a culvert. Input Representations exist to model linear and built-in features in size classes that would encompass most culverts. It is unlikely that Feature Analyst would be able to differentiate stone culverts from other types of culverts using a DEM or DEM-derived raster. If the majority of culverts in an area were modeled, it is possible that other GIS methods could be employed to determine the likelihood of a culvert being constructed of stone.

5 Conclusions

This project illustrated several ways in which LIDAR and Feature Analyst have great potential for processing and analyzing the significant quantities of LIDAR data available for the state of Kentucky. The research uncovered circumstances where each of the software's performance could be lacking. The sinkhole modeling demonstrated that it was important to clearly define the features to be modeled with Feature Analyst. Modeling for a group of features with highly variable size and shape characteristics was not ideal, but separating the features into more specific categories might be a fruitful strategy. The conclusion was that Feature Analyst had potential for modeling regularly shaped and sized features (cf. sinkholes).

The main focus of this project was modeling sinkholes, and less attention was focused on first and last return rasters, bare Earth DEMs, buildings, trees, and forest areas—features that were automatically extracted from the LIDAR data with minimal work on the part of the user. A great amount of time could be saved and innumerable projects could be built from those capabilities alone. Notably, while Feature Analyst was more unique in the options it afforded a user, some of the more basic functions of LIDAR Analyst were available for free with the LAsTools program², or the LAsTools Arc Toolbox. In the pilot projects, LAsTools was used to merge multiple LIDAR tiles into a single .las file because this function was absent from LIDAR Analyst. LAsTools was less user friendly than LIDAR Analyst, but it has an extensive online support community and ample customization options for users familiar with Python. Basic data viewing and management, point classification, and creation of DEMs were relatively straightforward in LAsTools. Performing more advanced extractions (i.e. buildings and trees) required a much more complex work flow. For projects that prioritize the extraction of buildings, trees, and forest regions, having LIDAR Analyst would be beneficial.

Before attempting large-scale projects with LIDAR and Feature Analyst, Condor's distributed batch processing software would have to be successfully installed. As noted earlier, our research team could not successfully install it. While Textron tech support and programmers were willing to help, they were never able to provide a solution. However, the batch processing capability built into LIDAR and Feature Analyst would be adequate for smaller projects.

² Available at <http://www.cs.unc.edu/~isenburg/lasools/>.

6 Works Cited

- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- Doctor, D. H., & Young, J. A. (2013). *An evaluation of automated GIS tools for delineating karst sinkholes and closed depressions from 1-meter LIDAR-derived digital elevation data*. National Cave and Karst Research Institute Symposium 2: 13th Sinkhole Conference. Retrieved from:
http://http://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=1156&context=sinkhole_2013.
- Jenness, J., Brost, B., & Beier, P. (2013). *Land Facet Corridor Designer: Extension for ArcGIS*. Jenness Enterprises. Retrieved from:
http://www.jennessent.com/arcgis/land_facets.htm.
- Zhu, J., Taylor, T.P., Currens, J.C., & Crawford, M.M. (2014). Improved karst sinkhole mapping in Kentucky using LIDAR techniques: a pilot study in Floyds Fork Watershed. *Journal of Cave and Karst Studies*, 76(3), 207–216.